Coordination for Energy

Clemens Grelck, Sebastian Altmeyer

Universiteit van Amsterdam

7th Workshop on Energy-Aware Computing (EACO'14)
Bristol, UK
September 10-11, 2014

What is Coordination all about ?

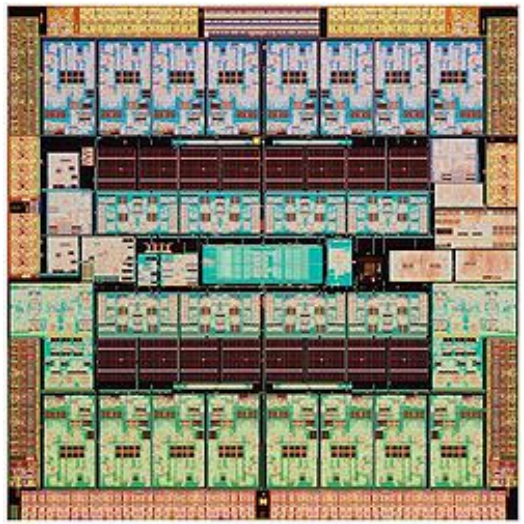
Coordination

Or, where we come from .....

The Previous Big Challenge: Multicore

... all of a sudden ....

... innocent computers turned into ...



... beasts

Coordination Principle: Separation of Concerns

Application engineering:

- implement problem-specific components in a familar environment
- reuse existing code written in legacy language
- make sure components are extrinsically state-free

Concurrency engineering:

- assemble (black-box) components to form parallel application
- define communication patterns and data dependencies
- leave mapping and scheduling details to runtime system

Coordination Language **S-Net**

S-Net principle: Separation of concerns

concurrency engineering ⟺ **application engineering**

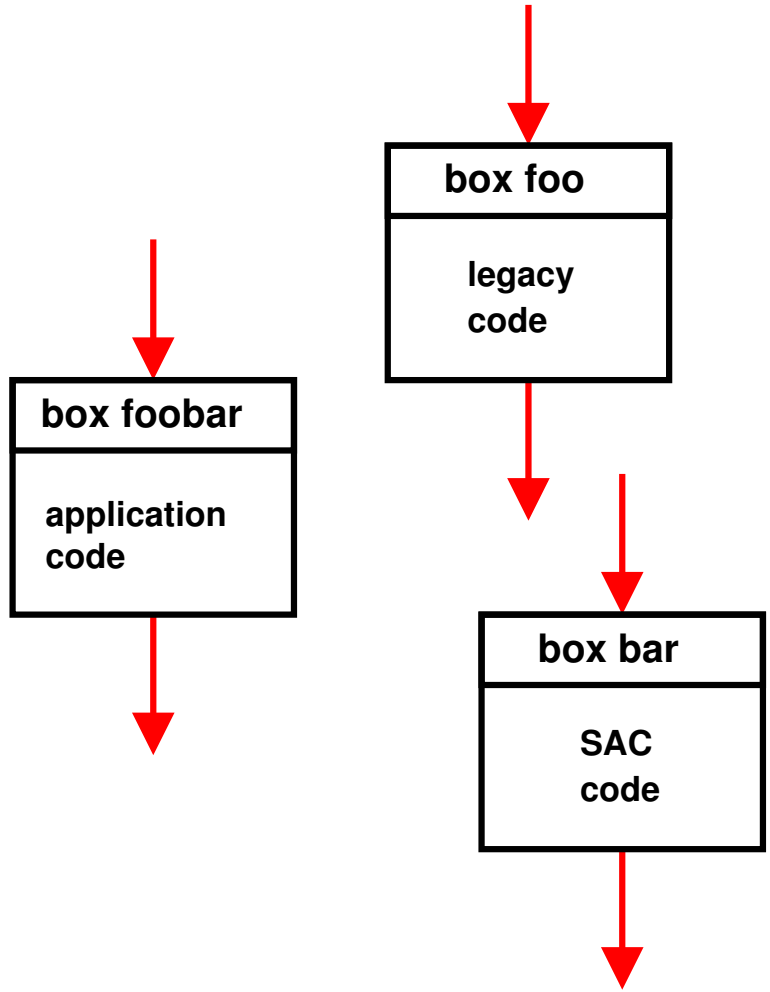
declarative coordination

+

stream processing

=

extreme decontextualisation



- Asynchronous components
- Single input stream
- Single output stream
- Context-free stream transformers

Streaming Networks of Asynchronous Components

Our principle: Separation of concerns

concurrency engineering ⟺ **application engineering**

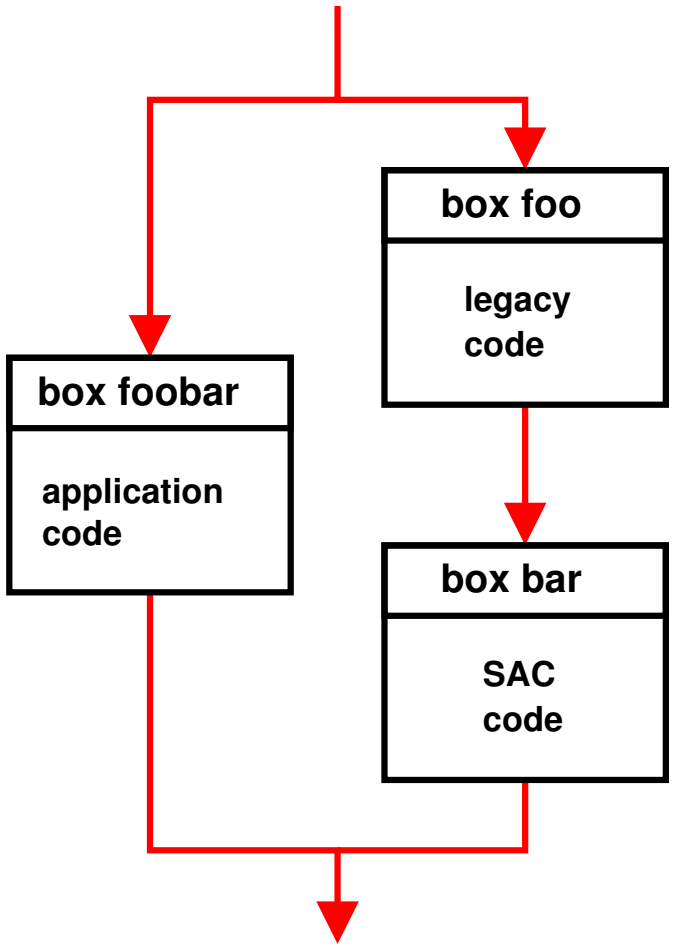
declarative coordination

+

stream processing

=

extreme decontextualisation

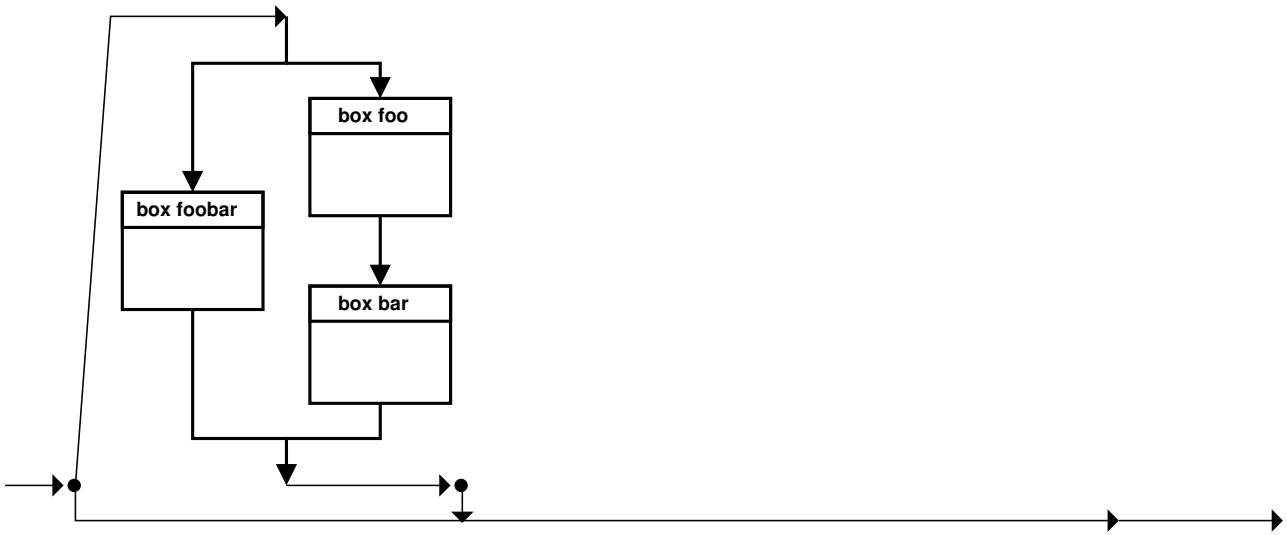


- 2 static network combinators:
  - static serial composition: `..`
  - static parallel composition: `|`
- Declarative network specification:

```
net cool
connect foobar | foo..bar ;
```

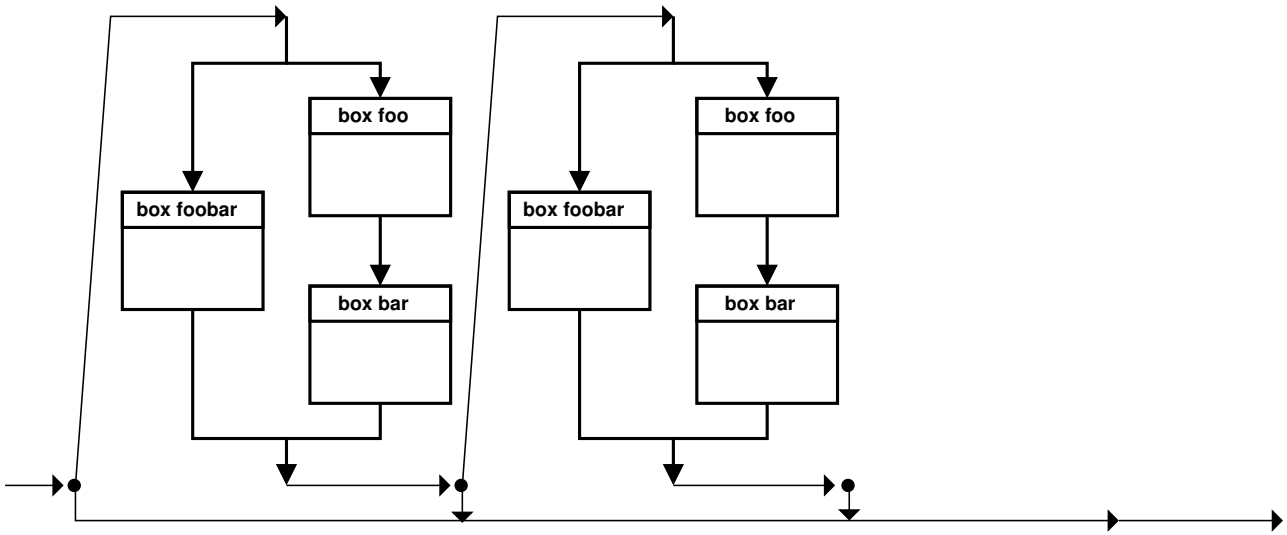
Two Dynamic Network Combinators

Data-driven serial network replication:



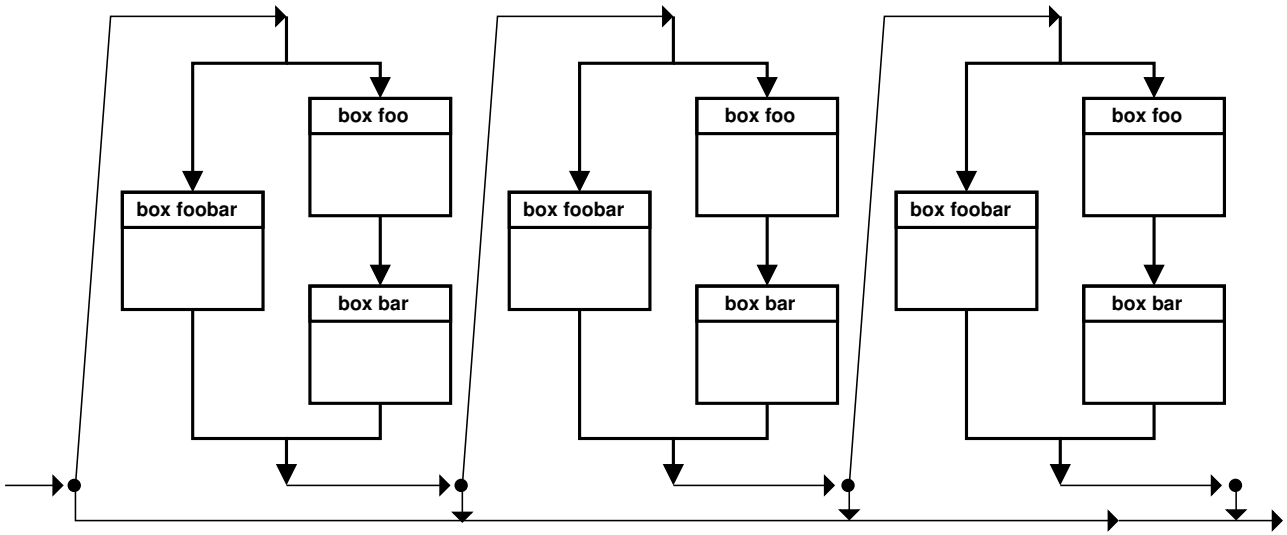
Two Dynamic Network Combinators

Data-driven serial network replication:



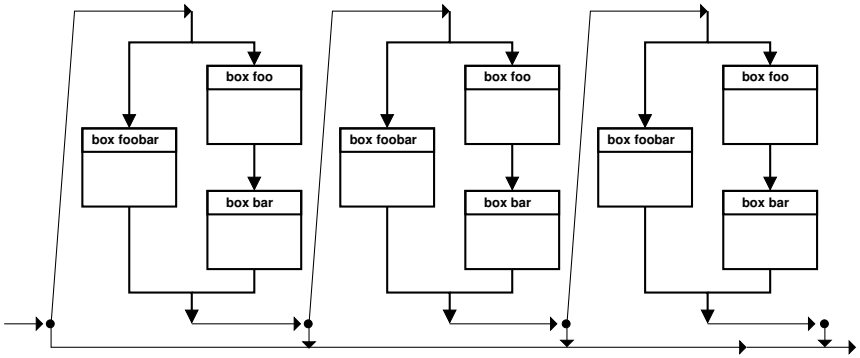
Two Dynamic Network Combinators

Data-driven serial network replication:

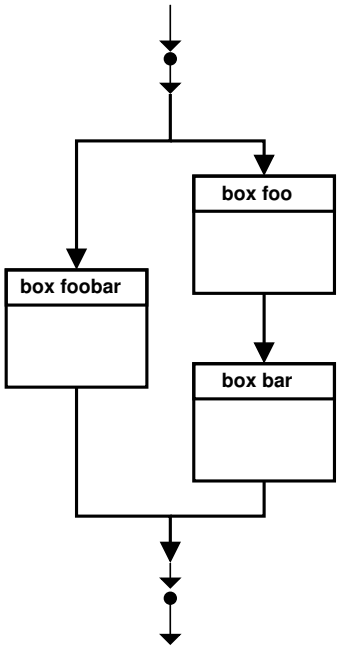


Two Dynamic Network Combinators

Data-driven serial network replication:

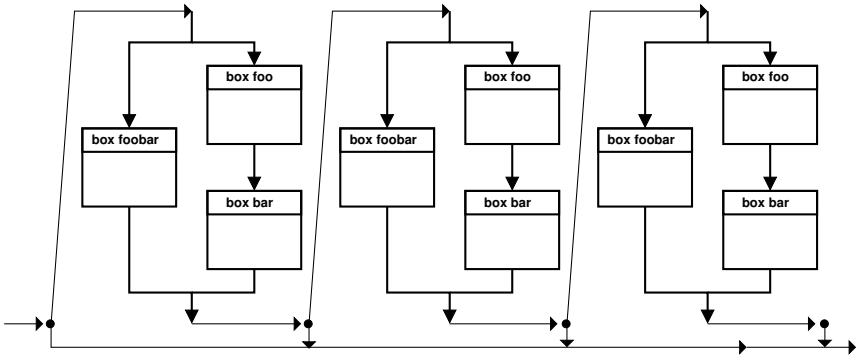


Data-driven parallel network replication:

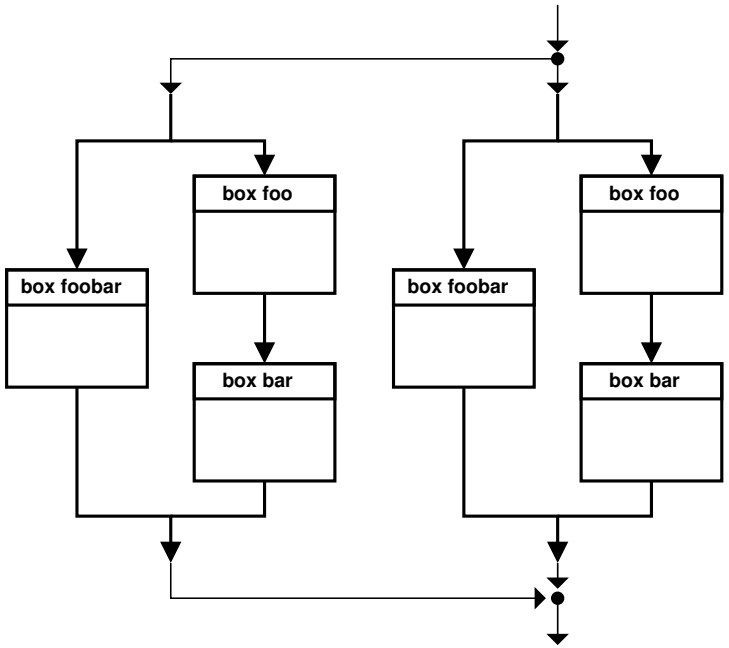


Two Dynamic Network Combinators

Data-driven serial network replication:

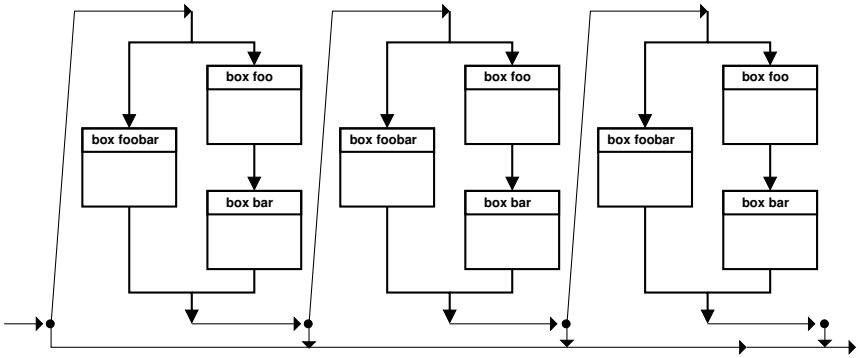


Data-driven parallel network replication:

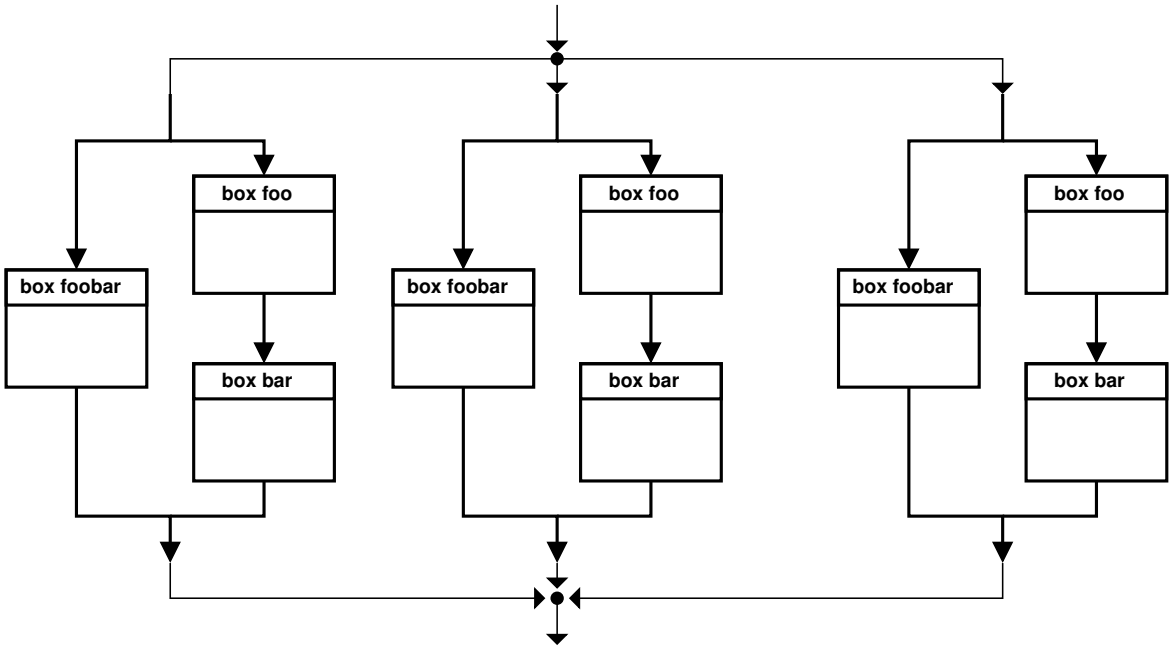


Two Dynamic Network Combinators

Data-driven serial network replication:



Data-driven parallel network replication:



Streaming Networks of Asynchronous Components

Our principle: Separation of concerns

concurrency engineering ⟺ **application engineering**

declarative coordination

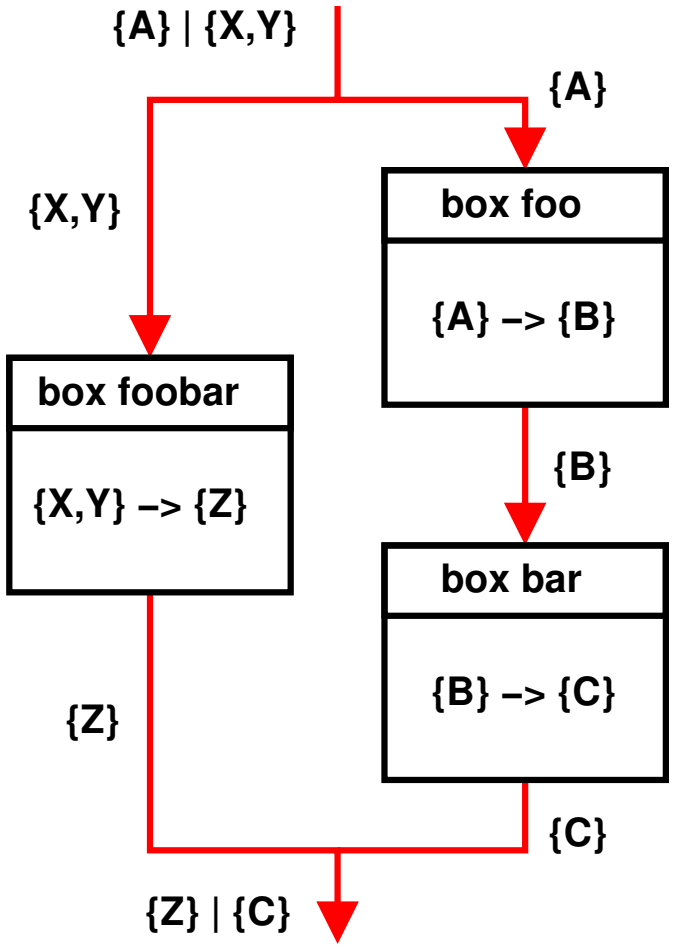
+

stream processing

=

extreme decontextualisation

- Box type signatures declared
- Stream types inferred
- Network integrity ensured
- Structural subtyping
- Type-directed routing

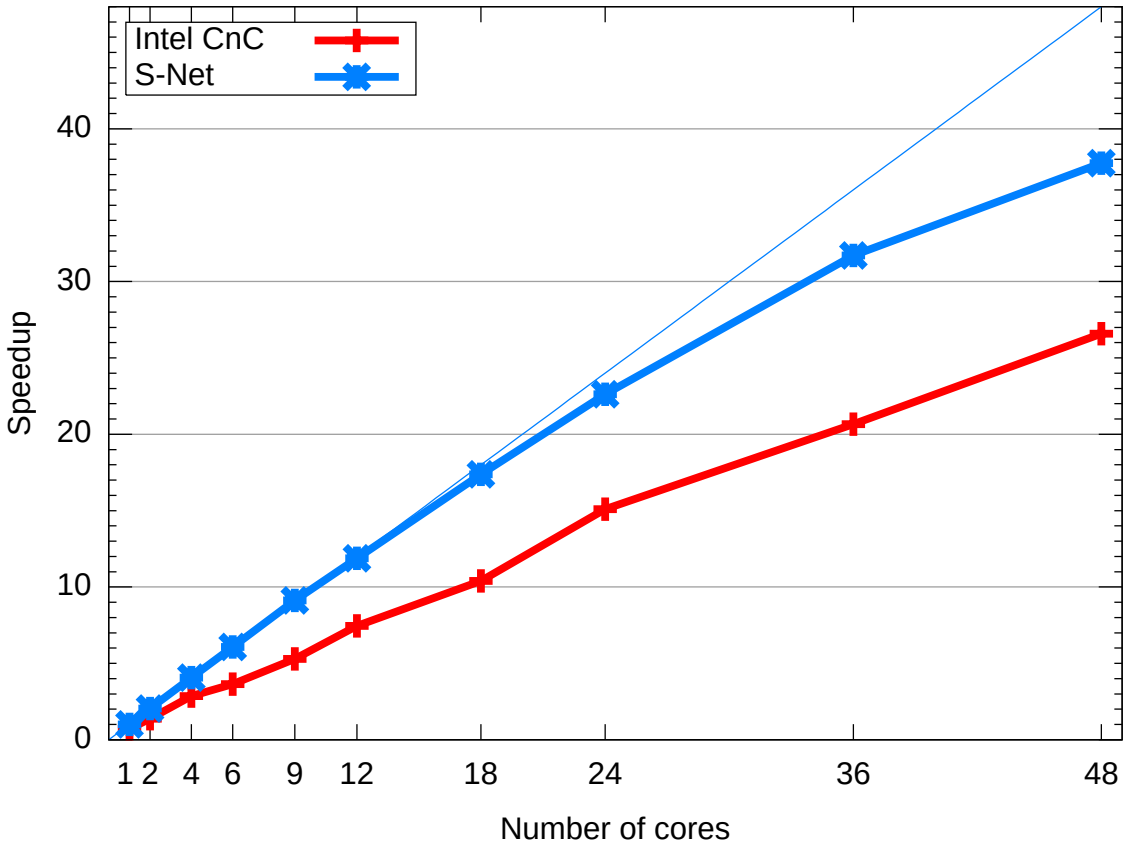


Experiment: **S-Net** on 48-core AMD Magny-Cours System

Machine:

- 4 AMD Opteron 8356 processors
- 12 fully-fledged cores each

Tiled Cholesky Factorisation:



What is Coordination for Energy all about ?

Coordination for Energy

Or, where we are heading for .....

Extra-Functional Properties of Boxes

Examples:

- Execution time
- Memory requirements
- **Energy consumption**
- Importance of computed results
- Delay bounds
- Internal elasticity
- ...

Extra-Functional Properties of Boxes

Examples:

- Execution time
- Memory requirements
- **Energy consumption**
- Importance of computed results
- Delay bounds
- Internal elasticity
- ...

Properties of Extra-Functional Properties:

- All possibly as functions of key parameters
- All more or less specific for different architectures
- Extra-functional properties = types ?

Extra-Functional Properties of Boxes

Where do extra-functional properties come from ?

- User annotations: mainly expert guess
- Tool inference: someone else's problem

Extra-Functional Properties of Boxes

Where do extra-functional properties come from ?

- User annotations: mainly expert guess
- Tool inference: someone else's problem

Can we trust extra-functional properties ?

- As much as you trust your programmers and tools ...
- EFPs are rather indicative in nature than exact

Extra-Functional Properties of Boxes

Where do extra-functional properties come from ?

- User annotations: mainly expert guess
- Tool inference: someone else's problem

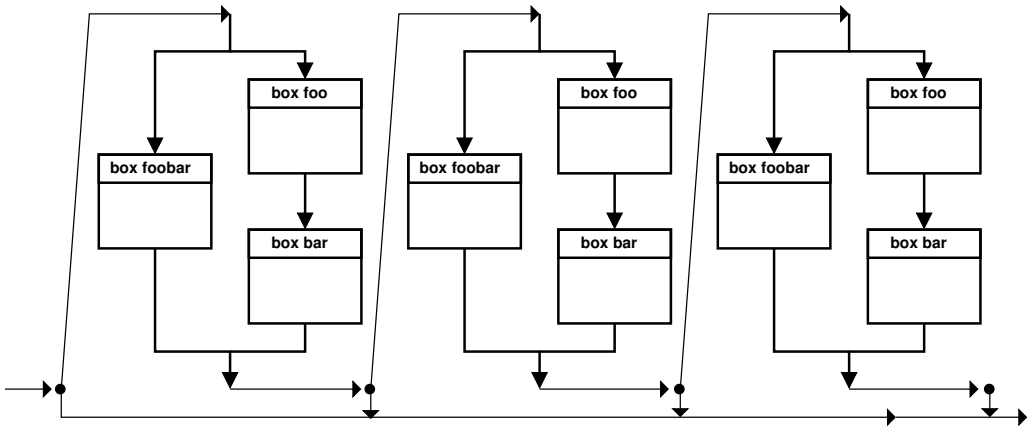
Can we trust extra-functional properties ?

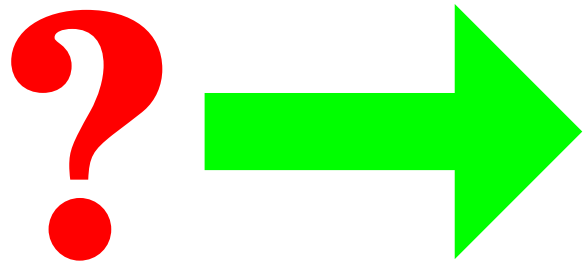
- As much as you trust your programmers and tools ...
- EFPs are rather indicative in nature than exact

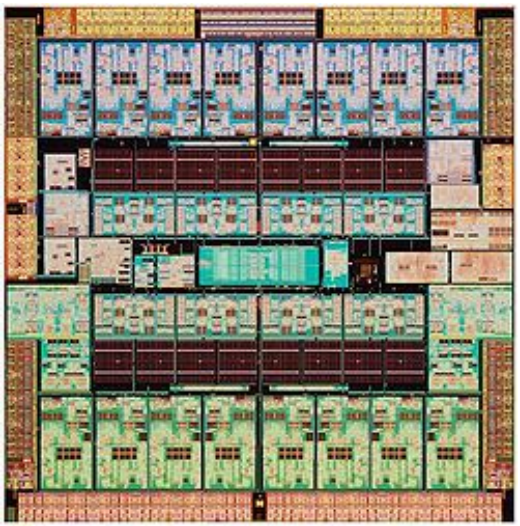
What can we do with extra-functional properties ?

- Not: predicting exact energy consumption
- Not: guaranteeing worst case execution time bounds
- But: improve scheduling and mapping decisions
- But: optimise runtime organisation for a given goal

Mapping and Scheduling







WWW: when to execute what where ?

Extending the Design Space

Alternative box implementations:

- Functionally equivalent
- Different extra-functional properties
- Could target different computing architectures (e.g. cpu vs gpu)
- Or different areas of heterogeneous SoCs (e.g. fat cores vs thin cores)

Mapping and Scheduling Options

Mapping:

- Choose the most energy-efficient (and available) execution resource for some task
- Compact component execution on nearby computing resources
- Actively manage computing resources (DVFS, partial shutdown)

Scheduling:

- Improve utilisation of computing resources
- "idle time is worst energy waste"
- Stretch computation in time rather than space (to some extent)

Monitoring

Monitoring

- Collect information about components at runtime
- "Learn" relevant properties
- Adjust annotations if needed

Conclusions and Future Work

Conclusions:

- Address energy concerns at the software system level
- Mix of static inference and dynamic decisions
- No strong guarantees, but best effort
- Effective parallel execution key to energy efficiency
- Remaining gain: under-utilised systems

Future work:

- Most of it .....